LISTING OF CLAIMS

1. (Currently Amended) A computer implemented method for rearranging a computer program comprising:

organizing the computer program logically into a plurality of blocks;

~~determining a critical section of the computer program;~~

constructing a dependency graph based on the organization of the plurality of blocks in the computer program;

determining a critical section included in the dependency graph;

detecting ~~recognizing~~ a portion of the plurality of blocks in the computer program that could be executed outside of the critical section; ~~and~~

inserting a plurality of dependency relationships based on the dependency graph between the plurality of blocks to cause execution of the detected ~~recognized~~ portion of the plurality of blocks in the computer program outside of the critical section; and

rearranging the detected portion of the plurality of blocks to outside the critical section that were inside the critical section based on the inserted plurality of dependency relationships.

2. (Currently Amended) The method of claim 1, wherein the plurality of blocks ~~a block~~ includes ~~a~~ computer program instructions.

3. (Currently Amended) The method of claim 1 further ~~comprises~~ comprising organizing the plurality of blocks in the computer program based on a node and a super block, wherein the node includes a plurality of blocks and the super block includes a plurality of nodes.

4. (Currently Amended) The method of claim 1, wherein the critical section included in the dependency graph ~~of the computer program~~ accesses shared resources.

5. (Cancelled)

6. (Currently Amended) The method of claim 1 ~~5~~ further ~~comprises~~ comprising adding a termination point to the critical section if a portion of the critical section is outside of the dependency graph.

7. (Currently Amended) The method of claim 1 further ~~comprises~~ comprising inserting an additional dependency relationship based on a direct dependency, an indirect dependency, or a shortest life-time dependency.

8. (Currently Amended) The method of claim 1 further ~~comprises~~ comprising scheduling to execute the plurality of blocks in the computer program based on the dependency graph, after rearranging the detected portion of the plurality of blocks to outside the critical section.

9. (Currently Amended) A computer implemented system for rearranging a computer program comprising:

a computer program organizer, ~~wherein the organizer organizes~~ to organize the computer program logically into a plurality of blocks;

~~a critical section determination module;~~

a dependency graph construction module, ~~wherein~~ to construct a dependency graph ~~is constructed~~ based on the plurality of blocks ~~organization~~ of the computer program; ~~and~~

a critical section determination module to determine a critical section included in the dependency graph;

a detection module to detect a portion of the computer program recognized outside of the critical section that could be executed by the processor; and

a dependency relationships inserter, ~~wherein the~~ to insert a dependency relationship is ~~inserted~~ between the plurality of blocks to cause execution of the detected ~~recognized~~ portion of the computer program outside of a critical section.

10. (Cancelled)

11. (Currently Amended) The system of claim 9, wherein the critical section included in the dependency graph ~~of the computer program~~ accesses shared resources.

12. (Currently Amended) The system of claim 11, wherein the dependency relationships inserter inserts a termination point to the critical section blocks if a portion of the critical section is outside of the dependency graph.

13. (Currently Amended) A system for processing a plurality of network packets comprising:

a network processor;

a network interface to control the transmission between the network processor and a network;

a shared resource accessible to the plurality of network packets;

a network processor program to process the plurality of network packets;

a dependency graph constructor to construct a dependency graph based on the network processor program; and

a dependency relationship inserter to optimize the network processor program by inserting a plurality of dependency relationships based on the dependency graph to rearrange the order in which the network processor program is executed.

14. (Currently Amended) The system in claim 13, wherein the dependency graph constructor further determines a critical section of the plurality of network packets and includes the ~~to the extent a~~ critical section in ~~is part of~~ the dependency graph.

15. (Original) The system in claim 13, wherein the dependency relationship inserter module inserts additional dependency relationships based on a direct dependency, an indirect dependency, or a shortest life-time dependency.

16. (Currently Amended) A non-transitory computer-readable storage medium ~~machine-accessible medium~~ that provides instructions that, when executed by a processor, causes the processor to:

organize a computer program logically into ~~based on~~ a plurality of blocks;

~~determine a critical section of the computer program;~~

construct a dependency graph based on the organization of the plurality of blocks in the computer program;

determine a critical section associated with the dependency graph;

detect ~~recognize~~ a portion of the plurality of blocks in the computer program that could be executed outside of the critical section; ~~and~~

insert a plurality of dependency relationships between the plurality of blocks to cause execution of the detected ~~recognized~~ portion of the plurality of blocks in the computer program outside of the critical section; and

rearrange the detected portion of the plurality of blocks to outside the critical section that were inside the critical section based on the inserted plurality of dependency relationships.

17. (Currently Amended) The non-transitory computer-readable storage medium ~~machine-accessible medium~~ of ~~method~~ claim 16, wherein the critical section included in the dependency graph ~~of the computer program~~ accesses shared resources.

18. (Currently Amended) The non-transitory computer-readable storage medium ~~machine-accessible medium~~ of ~~method~~ claim 16 further ~~comprises inserting~~ comprising instructions that insert a termination point to the critical section if a portion of the critical section is outside of the computer program.

19. (Currently Amended) The non-transitory computer-readable storage medium ~~machine-accessible medium~~ of ~~method~~ claim 16 further ~~comprises inserting~~ comprising instructions that insert dependency relationships based on a direct dependency, an indirect dependency, or a shortest life-time dependency.